درس معماری کامپیوتر

استاد: دکتر حمید سربازی آزاد

نیمسال بهار 1401

فاز چهارم پروژه

کمک پردازنده برای اعداد مختلط

اعضای گروه:

پویا یوسفی (98171223)

محمدرضا احمدی تشنیزی (98170646)

سجاد پاکسیما (98106286)

مهدیه ابراهیم‌پور (98170624)

فرمت دستورالعمل‌ها :

فرمت اول مشابه فرمت R فازهای پیشین است. یک Opcode شش بیتی و 6 رجیستر داریم که هرکدام با 4بیت Encode می‌شوند و دو بیت پایانی باقی‌مانده برای I/O درنظر گرفته می‌شود و فرمت ورودی و نحوه خروجی و ذخیره‌سازی را کنترل می‌کند.

فرمت دوم مشابه تایپ I فازهای قبل عمل می‌کند. یک Opcode شش بیتی و 4 رجیستر 4بیتی داریم و 2 واحد چهار بیتی برای Immediate استفاده شده که هرکدام برای مولفه‌های rs و immediate هستند. دو بیت پایانی صرف گرفتن ورودی و خروجی (I/O) می‌شوند و این فرمت برای دستورات Load, Store, Addi درنظر گرفته شده‌است.

نحوه کار:

در این فاز یک Register File برای اعداد مختلط اضافه شده‌است که شامل 16 رجیستر 32 بیتی است؛ از طرفی یک واحد حافظه شامل 256 خانه 8 بیتی یعنی 256 بایت در نظر گرفته‌ایم و یک واحد Forwarding برای Complex اضافه شده‌است که ورودی های ALU را با توجه به Hazard Detection کنترل می‌کند تا تمامی اعداد ورودی فرمت درستی داشته باشند.

نکات مربوط به :IDStage

ورودی‌ها داده‌هایی هستند که از Register File گرفته می‌شوند؛ خروجی‌ها نیز همان سیگنال‌های کنترلی لازم برای استفاده در استیج‌های بعدی هستند که در Control Unit اضافه شده‌اند و پس از انجام عملیات Decode خروجی از طریق IDStage خواهد بود.

رجیستر Complex که نوشتن روی آن انجام می‌شود، مشابه Pipelining بسته به Register dest بین rt , rd انتخاب انجام می‌شود و برای ورودی ALU نیز عملیات Converting خواهیم داشت؛ چراکه ورودی‌های Memory Stage و ALU و Writeback همگی از نوع قطبی هستند. درنتیجه در ID یک تبدیل فرمت ورودی انجام می‌شود و اگر ورودی بصورت نمایی بود، به قطبی تبدیل می‌شود و سپس به خروجی فرستاده می‌شود. این امر برای مقادیر val1 تا val4 انجام می‌شود. البته مقادیر val1 و val2 ممکن است خروجی مربوط به رجیستر فرمت پیشین باشند که این حالت نیز بررسی شده‌است.

مقدار ذخیره شده در Complex Writing برای دستور store به شرطی بررسی می‌شود که فرمت ورودی و خروجی یکسان باشد و به‌کمک Buffer به‌عنوان خروجی گزارش شود اما اگر فرمت Output یک بود، باید نوع قطبی به نمایی تبدیل شود و درغیر این صورت برعکس.

نکات مربوط به Forwarding و :EXEStage

ورودی‌های ALU بدلیل وحود Forwarding نیازمند Selecting هستند و علاوه‌بر Selectهای موجود در فاز قبل، 4 Select برای ورودی‌های ALU و 2 Select برای مقدار ذخیره‌شده درنظر گرفته شده‌است.

واحد :Memory

این واحد در mips_core قرار گرفته و ورودی همان مقادیر Bufferشده store valueهاست؛ خروجی نیز با write_back بافر می‌شود و داده‌های نوشته‌شده روی رجیستر در Complex، با توجه به فرمت خروجی (قطبی یا نمایی) Convert می‌شوند و به RegisterFile ریخته می‌شوند.

تست‌ها:

دستورهای این فاز به‌طور مجزا در قالب فایل‌هایی به زبان اسمبلی در فولدر test نوشته شده‌اند و به جهت این‌که وریلاتور بتواند این دستورات را تشخیص دهد، machine code این دستورات در فایل‌هایی با پسوند .mem قرار گرفته است.

برای مثال مستندات مربوط به دستور Addi را در بخش زیر مشاهده می‌کنید.

نکته: هر خط فایل اسمبلی موجود، یک بایت از حافظه را اشغال می‌کند.

```
1                   .text
2   main:
3           addi_c $t15, $t14, $t1, $t2, 2, 2
4                   addi_c $t13, $t12, $t3, $t4, 3, 3
5                   addi_c $t11, $t10, $t7, $t6, 4, 4
6                   syscall
```

فایل حاوی :Machine code

```
d4
4b
f8
88
d4
d3
70
cf
d5
da
e9
10
00
00
00
0c
00
00
00
00
```